Agile risk management using software agents

Odzaly, E. E., Greer, D., & Stewart, D. (2017). Agile risk management using software agents. *Journal of Ambient Intelligence and Humanized Computing*, 1-19. https://doi.org/10.1007/s12652-017-0488-2

Published in:
Journal of Ambient Intelligence and Humanized Computing

Document Version:
Publisher's PDF, also known as Version of record

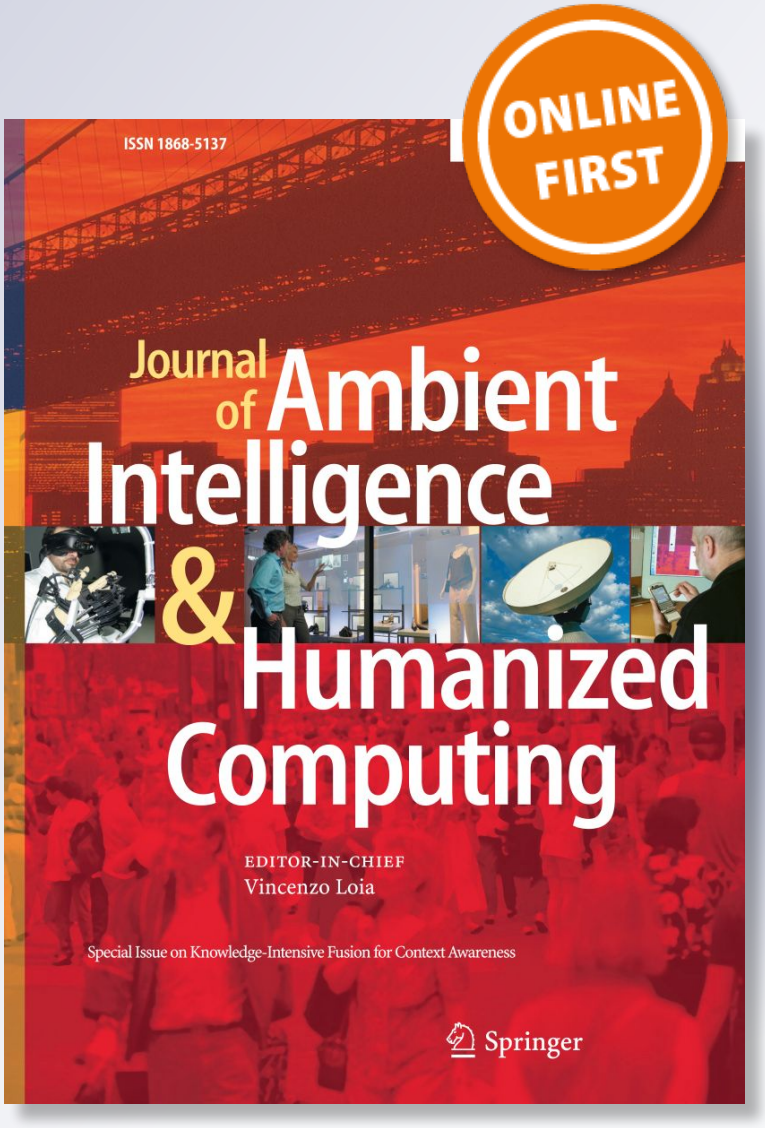
Agile risk management using software agents

Edzreena Edza Odzaly, Des Greer & Darryl Stewart

Journal of Ambient Intelligence and Humanized Computing

ISSN 1868-5137

J Ambient Intell Human Comput
DOI 10.1007/s12652-017-0488-2



Springer

J Ambient Intell Human Comput
DOI 10.1007/s12652-017-0488-2

ORIGINAL RESEARCH

Agile risk management using software agents

Edzreena Edza Odzaly[1,2] · Des Greer[1] · Darryl Stewart[1]

Received: 18 January 2017 / Accepted: 5 April 2017

Abstract Risk management is an important process in Software Engineering. However, it can be perceived as somewhat contrary to the more lightweight processes used in Agile methods. Thus an appropriate and realistic risk management model is required as well as tool support that minimizes human effort. We propose the use of software agents to carry out risk management tasks and make use of the data collected from the project environment to detect risks. This paper describes the underlying risk management model in an Agile risk tool where software agents are used to support identification, assessment and monitoring of risk. It demonstrates the interaction between agents, agents' compliance with designated rules and how agents can react to changes in project environment data. The results, demonstrated using case studies, show that agents are of use for detecting risk and reacting dynamically to changes in project environment thus, help to minimize the human effort in managing risk.

Keywords Software risk management · Agile risks · Agile projects · Software agents

✉ Edzreena Edza Odzaly
eodzaly01@qub.ac.uk

Des Greer
des.greer@qub.ac.uk

Darryl Stewart
dw.stewart@qub.ac.uk

[1] Queen's University Belfast, University Road, Belfast, UK

[2] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 77300, Merlimau, Melaka, Malaysia

1 Introduction

Risk management is recognized as a key process area in software development. Most risk management literature relates to heavyweight plan-driven processes and typically assumes that, for example, requirements have been agreed and signed off in advance of development. On the other hand, Agile Software Development uses an iterative approach to software construction, aimed at reducing development time, prioritising value, while improving software quality and inherently reducing risk (Cockburn and Highsmith 2001). This paper intends to demonstrate the idea of software agents to help manage risks in project development. This is achieved by using software agents to carry out risk identification, risk assessment and risk monitoring, the agents making use of data collected from the project environment. In the next section, we have highlighted the issues identified in risk management for an agile environment which further will be used as an input to the tool. In the approach used, the project manager has to define these elements: project goals, problem scenarios, consequences, risk indicators, project environment data as well as specifying risk rules using a predefined 'Rule template'. Next, the proposed Agile risk tool (ART) model is discussed, focusing on the development of the tool. This shows how the risk management activities are decomposed into agents, as well as how the interaction between agents is used to ensure that risks are appropriately managed. The use of a risk register is presented where a list of risks triggered in the project is displayed at a dashboard. The big advantage of the approach is that software agents can be used to detect risk and react dynamically to changes in agile project environment. To validate the approach some innovative case studies using student projects are described. Evidence is therefore provided for the feasibility and applicability of

Published online: 02 May 2017

the approach and finally some conclusions and discussion is given.

2 Research problems

2.1 Traditional risk management

Risk management in research papers is always acknowledged as being of utmost importance. To determine what is needed we used existing work (Odzaly et al. 2009) on an investigation of the barriers to risk management. The results of that investigation concluded:

- That there is no standard or commonly adopted risk management process and/or tool being used in software development situations.
- That Risk Identification was the most effort intensive process and additionally 30% agreed that risk monitoring is most difficult and needs more effort.
- That the biggest barrier was that visible (and tangible) development costs get more attention than intangibles like loss of net profit and downstream liability.
- Despite the acceptance that risk management methods enhance system development, nonetheless little support is to be found on the provision of these methods (Ropponen and Lyytinen 1997). It has been argued that the methods of managing risk in software development are not comprehensive as they deal with specific types of risk (Bandyopadhyay et al. 1999). Besides, despite many well known risk management approaches having been introduced, risk management was still reported as not being well practiced (Ibbs and Kwak 2000; Pfleeger 2000). As reported in Bannerman (2008) the most common risk management approaches found in the literature highlight practices such as checklists, analytical frameworks, process models and risk response strategies. Many researchers have conducted research in tailoring risk management, providing various approaches. However, only a few studies have reported integration of risk management with contemporary software development (Nyfjord and Kajko-Mattsson 2008).

2.2 Risk issues in Agile software projects

It is clear that people issues are the most critical in agile projects and that these must be addressed if agile is to be implemented successfully (Boehm and Turner 2005). Indeed, one of the most important success factors in an agile project is individual competency (Cockburn and Highsmith 2001). Additionally, estimation of effort is a consistent challenge in agile development work, especially when it is done for the first time (Deemer et al. 2010) and there are issues with agile skills and personnel turnover, as well as job dissatisfaction (Boehm and Turner 2003; Melnik and Maurer 2006; Melo et at. 2011). In Scrum individual motivation is very important and influences how diligent team members are; for example in attending daily scrum meetings (Hossain et al. 2009). Recognising non-compliance with established practices can provide early signs of risks e.g. low morale expressed during the daily meeting or avoiding discussing problems when behind schedule (Lindvall et al. 2002).

Due to the fact that agile methods depend a lot on the credibility of the people involved in the projects (Cockburn and Highsmith 2001; Nerur et al. 2005) as well as their motivation in applying the agile practices (Layman et al. 2006; Conboy et al. 2010), most issues encountered relate to the people and the practices involved. This echoes one of the values in agile manifesto i.e. “individuals and interactions over processes and tools” (Fowler and Highsmith 2001). This implies that not having the right people doing the right process will be a source of risk.

Cho (2008) developed some research work on issues and challenges of agile software development with Scrum. The research work mainly aims to provide guidelines to assist the companies to avoid and overcome barriers in adopting the method. An in-depth study was conducted between two different companies using various qualitative data collection methods. The study presented various common categories of issues and challenges in agile projects, among which the following points are discussed in Table 1.

Chakradhar (2009) has outlined the common pitfalls in agile projects in which, he proposed that it is vital for the project manager to understand those pitfalls in order to reduce risks. Among the important aspects discussed are: failure to provide sufficient training in agile methodologies; unfamiliarity of the project manager with agile methods; poor involvement from the Product Owner; the team practicing ‘single expert’ with no knowledge sharing as well as having passive team members.

Cockburn and Highsmith (2001) highlighted that one of the most important success factors in a project is individual competency. They emphasize the qualities of people involved in the project, where good people will complete the project while if team members do not have sufficient skill, no process can compensate for their deficiency. This is also supported by Boehm and Turner (2005) where people issues are the most critical and it stated as very important to address them before adopting and integrating agile practices into a project. That paper presents a list of barriers to using agile methods successfully and among the significant issues highlighted with respect to people are their roles, responsibilities and skills as well as their ability to predict and be knowledgeable.

Table 1 Categories of issues/challenges in agile projects (Cho 2008)

Categories	Issues found
Human resource	Formation of team, where team was being organized without considering their necessary skills and knowledge
	Multiple responsibilities where one team member is responsible for many tasks
	Some developers are not aware of the benefits of applying agile methods, and so are reluctant to apply agile practices
	Lack of accountability where team members do not take responsibility for any delayed task coupled with a lack of supervision
	Collaboration between team members is difficult, especially when they are not located together
Structured development process	Scrum meetings; daily scrum, sprint planning and sprint review meetings are sometimes inefficient where they are being held too often, or taking up too much time or setting up the meeting is difficult
	Difficulties in estimating project work on legacy code
Environmental	Poor customer involvement and unclear product requirements
	Individual contribution is not recognized and no guidelines exist in determining accurate measurement of individual performance
Information systems and technology	A lack of communication between team members that are co-located causes duplication in resolving problems
	Newly hired team members tend to create code errors due to unfamiliarity with the software

Deemer and Benefield (2006), discuss common challenges in Scrum. One of the challenges put forward is the ability of a team to provide estimation of effort in their development work especially when it is done for the first time. Most teams fail to deliver the tasks committed to due to poor task analysis and estimation skills. When this happens, the team tends to extend the duration of the sprint rather than learn to do the correct estimation. This can cause problems in achieving a sustainable pace since the team will not be able to work reasonably due to delay in completing other tasks in the project.

Having a team member that is an agile sceptic, meaning, they are opposed to agile methods, can have a huge impact to the team as a whole. This is due to the fact that an agile team relies heavily on trust and sharing of tacit knowledge to support important practices like pair programming and shared ownership (Boehm and Turner 2005). Melo et al. (2011) presents an unusual result with regard to the relationship of pair programming to tasks and motivation. Surprisingly, whether the tasks are too easy or too complex, may influence the motivation to work in pairs.

Another important practice in agile process is the collective code ownership. The study results discussed in Layman et al. (2006) indicates that collective code ownership provides benefits in terms of knowledge sharing within the team. However, the disadvantage of this is that the team will tend to choose the fastest solution ignoring its quality, assuming that they are not the only one who is responsible for the quality or otherwise of the code. By not assigning task ownership for a piece of work, this could demotivate the team in writing code that conforms to the standards or required quality levels. Other problems discovered are working at a sustainable pace and accuracy of estimation. These problems are due to the situations like having aggressive datelines, having to work overtime as well as underestimation of completion of time.

Nelson et al. (2008) introduced a risk management technique that can be adopted in agile processes. The paper also provides an argument in relation to agile being risk driven in that it implicitly manages risk in the process. One of the implicit techniques used is to prioritize tasks at the start of iteration. However, simply placing higher priority to a high risk task is not considered as managing risk. It does reduce the risk to the project if the associated task is executed earlier, but until the risk is resolved or no longer applicable, the task needs to be monitored for the risk, and action taken if necessary. When identifying the right risk for the task and analysing it is not done properly, presenting an appropriate plan to mitigate the risk is difficult (Williams et al. 1997; Ahmed et al. 2007).

3 Solution approach

As a result of the issues identified, there is a strong motivation to improve the management of risk in agile projects without reducing agility in projects. In reality, contemporary risk management should be able to be integrated into the agile process to support decision making. This includes taking into account human factors such as developer skills and ability as well as their behavior in performing tasks. Due to the fact that Agile relies heavily on the competency of the people involved, therefore we converted these issues to risk factors i.e. situations or events that may cause a loss to occur and therefore that we need to monitor in a project as shown in Fig. 1. In order to establish the basis of the ART model in relation to risk management, a model called Goal-driven

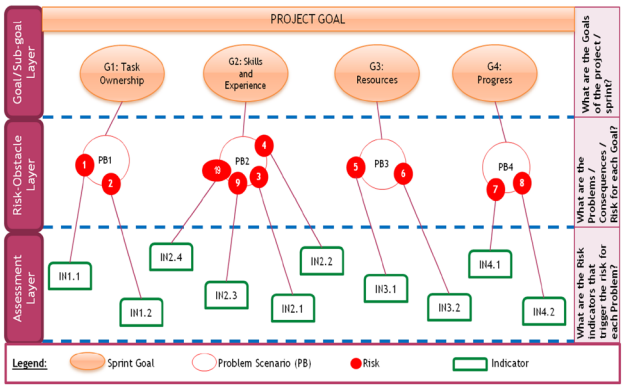


Fig. 1 Agile risk tool (ART) GSRM Model

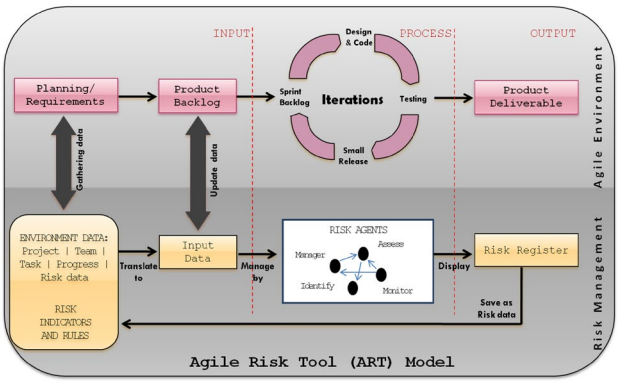


Fig. 2 Agile risk tool (ART) model describing the application of risk management in Agile environment

software development risk management model (GSRM) proposed by Islam (2009) was reused. This model is used to specify the input for the project, which consists of the type of risks and the risk indicators as well as the environment data which can be used to identify the risks for the project. Thus the issues discussed earlier are transformed into a set of sprint goals. These will later be used to define the risks and their indicators thus allowing risks to be monitored continuously.

The following section will discuss the development of the ART model and how it is used in dynamically managing risk.

3.1 The agile risk tool (ART) model

The development of the ART model started with the establishment of a view of how risk management may apply in an agile environment. Figure 2 below depicts an overview of the resulting model.

3.1.1 Input

The model represents how risks are gathered and managed throughout an agile project. During the *Input stage*, the agile process begins with planning and requirements gathering. At this stage, while preparing the project, at the same time, the gathering of risk data can commence. Requirements in agile processes are most often represented as user stories. These are textual descriptions that contain the customer's specification of needs for the required system. A product backlog is a subset of these requirements that will be selected from based on priority.

The environment data used contains:

- A project in this context is a set of user stories, the membership of which is not fixed at any point of its lifetime. Each project relates the unique project name of the project, a set of goals for the project, when it started and when it ended.
- A team is a set of persons where each person consists of a set of attributes describing the person. Each team is working to achieve the goals of the project. For each team member there is specific information, for example on the type of skills that the team member possesses and also their levels of expertise in defined skills, stated as an integer;
- User stories are divided into tasks. A task refers to a textual description of the task associated with the estimated hours of completion, the name of the person responsible for the task and the progress for the task;
- Progress refers to additional information on the progress of a specific task as reported by the person responsible for the task. This includes information on attendance of the team member in the daily scrum meeting and whether progress or an impediment is reported for the task;
- Risk data represents information on risk captured by the tool. The information includes the name of the risk, its severity, the owner of the risk, location of the risk as well as the date the risk is triggered and resolved.

The risk indicators and rules refer to a set of predefined risk factors brainstormed by the team at the early stage of the project and encoded as rules (this will be further discussed in the next subsection). The risk indicators contain a textual description indicating a threshold or state that will trigger the risk. One example might be where a high priority task is selected in the sprint by a developer with too low a predefined skill threshold. Rules contain a list of conditions for an event encoded into IF/THEN statements. Later, this information is stored in the rule engine. Input data refers to a set of collected data from the environment and translated into a set of templates readable by the tool.

During the Process stage, the project proceeds as iterations which include sprint backlogs, design and code, testing and small releases of the product requirement. Iterations contain are time-boxed into fixed length durations of development. Risk agents (or ART agents) will manage the risk based on the input data defined earlier. This risk process is autonomous, where software agents; identify, assess and monitor risk based on the input data from the environment. Once any risk is triggered, risk data will be displayed in the Risk register. Any changes or updates to the environment will affect the risk data (whether or not the risk is flagged up).

At the *Output stage*, the final risk data can be obtained after the delivery of the product and during a Sprint review meeting. The risk register provides a view of all identified risk data. At the end, the data displayed in the Risk Register can be recorded and saved in the risk data repository where this information can be used to plan future projects.

The model has been demonstrated further and used as part of the work in (Odzaly et al. 2014). This is where the ART architecture proposed was demonstrated in order to explore the application of risk management in agile applications. This paper complements that in focusing on the development of ART agents used at the Process Stage.

- The development of ART Agents

One way to move towards automation is to give software agents responsibility to identify, assess and monitor risk. These agents ideally should be able to autonomously react to environmental changes, where the environment in this case is the software development environment, including the set of tools being used.

In order to reduce barriers in risk management application, a lightweight risk management approach is needed. The newly proposed approach includes three main steps in risk management; risk identification, risk assessment and risk monitoring. The rationale of doing so was twofold (1) to develop a realistic and acceptable risk management process that can fit into the agile methods (2) an empirical study (Odzaly et al. 2009) confirmed the most complicated steps in managing risks were risk identification and risk monitoring. In addition, prior to this section evidence is established that contended that risk management was difficult mainly due to the required human effort. Given this, the aim is to substitute some of the human involvement with autonomous software agents with the goal that these could manage risk and minimize the need for manual input. Automated agents can therefore help ease the work load in managing risk, specifically in identifying, assessing and monitoring risk.

Decomposition of risks into activities is commonplace. One example discussed in (Kontio 1997) used

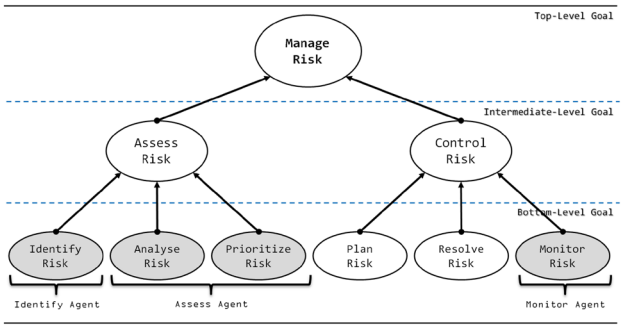


Fig. 3 Risk decomposition graph for the Agile risk tool (ART) agents of four risk management activities

decomposition of risk into conceptual elements like risk factor, risk event, risk outcome, risk reaction, risk effect and utility loss. More recently a top down goal decomposition technique is described in Bresciani et al. (2002) and Dardenne et al. (1993). Indeed Boehm's tutorial on risk (Boehm 1989) decomposes risk management into activities. In this work the category or type of agents used was derived based on initial agent goal decomposition as shown in Fig. 3, based on Boehm's work.

The generic aim of this work is to find ways of lowering the barriers to application of risk management. One of the objectives is to use the agents since agent behaviour is more adaptable and can act on behalf of the project manager of the agile project. In this case, some of the effort of the project manager is replaced by agent execution such that they will react automatically according to their own goals. In identifying goals for the agents, the top level goal is started in order to apply risk management in software development project, particularly in agile projects. This goal is further decomposed into two intermediate sub goals; assessing risk and controlling risk. These sub goals are then decomposed into six smaller sub goals; identify, analyse, prioritize, plan, resolve and monitor. As a result of the decomposition of the goal, agents were assigned based on the smallest sub goals which supported the top level goal. Since the most effort intensive steps identified earlier were identification and monitoring, for the meantime, both sub goals were selected in addition to analyse and prioritize goals as highlighted in Fig. 3. Note that here that only the bottom level goals are engaged; the assumption being that top and intermediate level goals might have largely a controlling function but nonetheless have their own goals on how lower level agents should interact.

Further ART agents were developed for this work as four agents; manager agent, identify agent, assess agent (combines analyse and prioritize goals) and monitor agent. This is depicted as in Fig. 4 that shows the interactions (communications via passing message) between manager agent and the identify, assess and monitor agents. Depending on

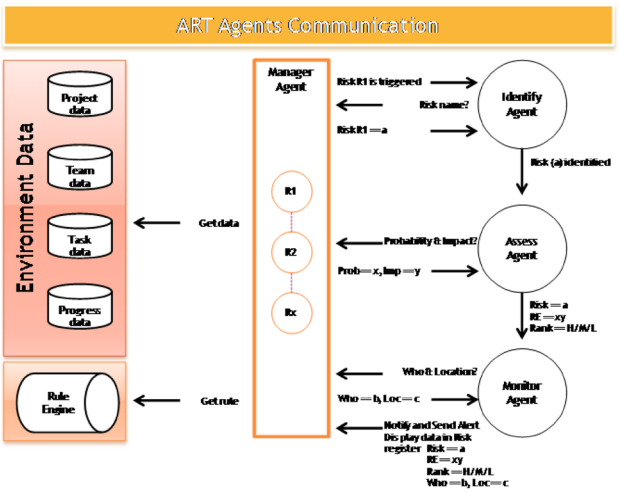


Fig. 4 The communication between the ART agents and how they interact within the environment data and rule engine

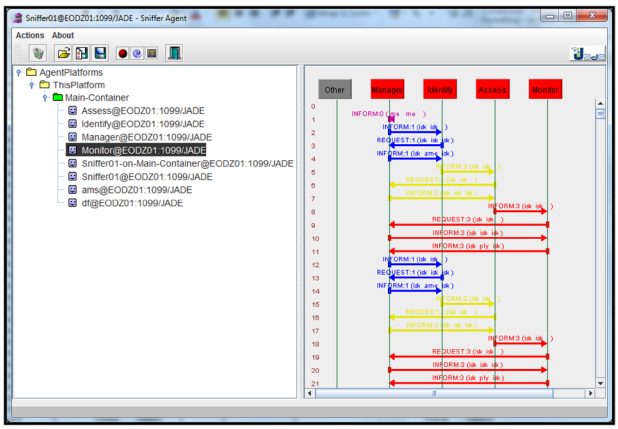


Fig. 5 Sniffer Agent

the data from the environment, the agents react to detect risk dynamically through rules execution, where rules are invoked from the rule engine. The ART agents' communication is described further as below.

There are four ART agents and each of them has a designated goal assigned to them. The goal and purpose of each of these is discussed below.

- *Manager Agent* acts as an intermediary between the other three agents. It manages and executes rules, gets data from the Environment and notifies Identify agent if any risk is triggered.
- *Identify agent* is notified if any risk is triggered. It requests from the Manager agent what risk has been identified and notifies the Assess agent.
- *Assess agent* is invoked by the Identify agent and its goal is to estimate the risk exposure (RE) of the identified risk where RE = probability (P) × impact (I). The identified risk will then be ranked as high, medium or low and the monitor agent is notified to take subsequent action.
- *Monitor agent* is invoked by the assess agent with some data: RE and rank of the identified risk. The monitor agent will establish the location of the identified risk along with the owner of the risk. This data is then displayed in the Risk Register which later can be recorded and saved in the risk data repository.

3.1.2 Process

At the *Process Stage*, the ART agents will monitor the risk by acknowledging any rules or risk indicators triggered as informed by the ART template. The ART agents will initiate communication between them. Messages are passed according to request and each agent will notify another agent in prompting any further action to be taken. An example of the ART agents' communication was introduced earlier in this chapter (Fig. 4).

Figure 5 show an interaction between the ART agents starting when a risk is triggered. The figure shows the agents passing message using the Sniffer agent in the JADE platform. True to its name, sniffer agent is a purely java application that tracks messages in the JADE environment. It is useful when debugging the agent behaviours and for analysing message passing using in the sniffer GUI (Bellifemine et al. 2007).

Rules and the environment data are dynamically editable. In the event where changes need to be made, one can modify the environment data (which has been translated into the ART template earlier) as well as the risk rules and indicators using the provided main screen area. On the other hand, when developing possible risks associated with rules and risk indicators, one might find the environment data used to be insufficient to detect certain risks. In some cases, a small change in collection of the environment data would allow defining or detecting more risks. For example, adding the information on developer's skill will allow monitoring the developer's programming capability especially in completing high priority task. An example of a rule syntax that can be used is, "IF the developer skill level is 'low' AND the developer involved with a 'high' priority task, THEN there is probability a risk of the task cannot be completed on time because of the developer's poor programming skill".

ART agents will react dynamically to input data, process the input by assessing any risk triggered and produce a risk result in the Risk Register.

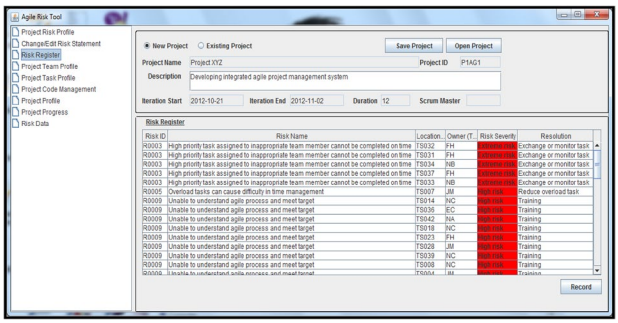


Fig. 6 Agile risk tool—Risk Register

3.1.3 Output

The idea of a Risk register has been defined by Willams (1994) who states that "the risk register has two main roles. The first is that of a repository of a corpus of knowledge… The second role of the risk register is to initiate the analysis and the plans that flow from it". While (Patterson and Neailey 2002) reported that very few development and construction of risk registers although it is commonly used in Risk Management. As such, risk register developed in this work can represent as a risk dashboard in which one can see a list of risks triggered by the ART agents. The Fig. 6 shows an example of risk register used as the visualization of output in this tool. An overview of the main algorithm is described below:

Input

env_data = Environment_data items (project, team, task, progress, risk) as sensed from tools used. Here we will refer to a generic type env_data.
risk_indicator[ ] = sets of threshold values for each type of environment data. There is a 1:N association between each env_data type and risk_indicator. Each risk indicator represents a measurement than can be sensed in the environment for that environment data item.
risk_rules[ ] = User defined ruleset defining for each risk_indicator how to react when the value of env_data meets/exceeds/falls below risk_indicator. There is a 1:N association between risk_indicator and risk_rule[i].

Output

risk_register[ ] = set of risks where risk is a textual statement and a conceptual value (high, medium, low)
Dashboard = display of risks

Functions

env_data_onChange(): This function exists for each env_data item and is triggered when the sensed value has changed
checkRiskCondition(env_data) is a function that compares the value of env_data with the appropriate indicator and returns a set of applicable rules from the ruleset, risk_rules[ ]
applyRule() calls any necessary actions (e.g. emails alerts)
updateRegister() appends or updates risk in the risk_register[ ]
updateDashboard() notifies the dashboard of new/modified/deleted risks.

Algorithm for each env_data:

```
env_data_onChange()
    BEGIN
      applicableRules :=
          checkRiskCondition(env_data)
      ForEach rule in applicableRules
          BEGIN
              applyRule()
              updateRegister()
              updateDashboard()
          END
    END
```

4 Case studies

In order to show that the suggested approach is feasible case studies were conducted using student project artefacts.

4.1 Case study methodology

This work aims to answer the following Research Questions (RQs):

1. RQ1: Is it possible to support risk management in agile projects using existing SE environment data to overcome barriers in the application of risk management?
2. RQ2: Can data collection be conducted with minimal intrusion and effort?
3. RQ3: Can software agents coupled with a rule engine provide a means to automate risk management in agile projects using data from the software development environment?

In order to address these, case studies were used using the proposed solution approach and tool support. The choice of case study was based on the fact that the development of the approach was rather new and exploratory involving a large number of variables; thus a flexible and natural method was needed. Easterbrook et al. (2008) declared exploratory case studies to be suitable for preliminary investigations of some phenomena to develop new hypotheses and build theories, and confirmatory case studies to assess the existing theories. Case studies can provide a deeper understanding on the subjects under study and the results obtained are valuable as well as contributing to the body of knowledge (Kitchenham et al. 1995).

Easterbrook et al. (2008) promotes case study research where it uses the purposive sampling, thus depending on the nature of the research objectives. A case study is suitable when research is required in order to gain deep insights into chains of cause and effect. Furthermore, exploratory investigations are appropriate where there is little control over variables in the study. It started with the investigation into issues and problems in risk management in agile projects in order to gain deeper understanding of the phenomena followed by a proposed solution approach. Later, the solution approach is validated using a developed prototype tool using the case studies labelled CSA and CSB. The first case study, Case Study Alpha (CSA) was considered necessary to explore the problem domain and was preliminary in nature, mainly intended to develop the Research Questions above. The second case study, Case Study Beta (CSB) was conducted as a confirmatory case study and was used to assess the existing theories and results developed from CSA. However, both case studies aimed to provide validation of the solution approach and the tool support with improvements being made in CSB, based on the lessons learned from CSA. Mixed methods were used including (1) an informal interview with the Product Owner to validate results from the prototype tool and (2) artefact or archive analysis was done to understand compliance with agile practices in the team as well as to demonstrate the outcomes generated from the prototype tool. The artefact or archive analysis refers to the investigation of project data which includes the Agile Project Management tool spreadsheet they used to collect data, minutes of the meeting and SVN repositories in order to identify patterns in the behaviour of the development team.

The case studies were carried out on groups of students who were tasked with a software project and used their data as an input to the proposed approach. Many practitioners in agile projects claim that agile methods inherently reduce risk (Boehm and Turner 2005; Cohn 2005) but to the authors' knowledge very little research has been done to confirm this or in relating these two areas. However introducing something new in an organization is difficult and likely to be costly. Menzies et al. (2009) use the term 'data drought' to describe the situation where there is an unwillingness from organizations to share their operational data due to, among other factors, business sensitivity associated with the data. Given the difficulties of obtaining industrial data coupled with the ready availability of student project data in a university setting, student group projects were used in order to demonstrate and validate the approach. On the other hand, the use of student project data has strengthened the approach due to the access to a unique data set which is not be available in any other setting. This includes the access to data for a set of parallel agile teams all carrying out identical sets of user stories. This scenario would be very difficult to find or engineer in a similar study elsewhere, and virtually impossible in an industrial setting.

4.2 Case study design

The study used data from a final year undergraduate honours course in Agile Methods, taught at Queens University Belfast in the years 2011 and 2012. In the theoretical part of the course, students received lectures on general agile development practices with an emphasis on Scrum. During the course, students were required to build a large software artefact using Microsoft.NET technologies using an industrial strength environment adopting both agile project and software engineering practices. This includes applying important Scrum project management practices such as Pair Programming, Test Driven Development, Release and Iteration Planning and Refactoring in their software project. The first case study was developed in 2011 involving 38 undergraduate students, assigned into six groups with six or seven developers each. All groups were required to develop the same product requirements. The first case study was developed in 2011 (CSA) with groups labelled Alpha1 to Alpha6 (Alp1–6). The second case study was developed in 2012 (CSB) and involved a total of 56 undergraduate students with eight groups, Beta1 to Beta8 (Bet1–8) each group consisting of five to eight developers. All groups were given the same product requirements as in the first case study. However, due to some missing data from group Bet8 this group was dropped from the study, leaving only 48 undergraduate students involved in this study. The projects in both case studies involved two sprints, SP1 and SP2 which respectively had at least 10 to 15 working days. Before the start of the project, the Product Owner, a role played by a member of academic staff, introduced the Product backlog items which consist of a list of prioritized user stories. Thereafter, it was each group's responsibility to deliver these to the satisfaction of the Product Owner. The Product Owner and students were able to communicate regarding any arising issues in both sprints; therefore the groups have been supervised throughout the development process.

Table 2 The summary of collected environment data from the student projects in Case Study Alpha (CSA) and Case Study Beta (CSB)

No.	Environment data Attributes	Value	Used	Repository
1	Project ID	Project unique number	✓	Project data
2	Project status	Not started, in progress, completed	✓	Project data
3	Team name/ID	Team member name or unique number	✓	Team data
4	Role	ScrumMaster /developer	✓	Team data
5	Total No. of role assigned	1 or 2 roles	✓	Team data
6	Total No. of project involved	1 or more projects	✗	Team data
7	Team skills	Programming (C#)	✗	Team data
8	Agile experience	True/false	✓	Team data
9	Agile level	Very good, good, average, poor, very poor	✓	Team data
10	Skill level	5 (highest skill) to 1 (lowest skill)	✓	Team data
11	Task Name/ID	Task name or ID	✓	Task data
12	Task priority	High, Medium, Low	✓	Task data
13	Paired by	Paired or Not Paired [“ ”]	✓	Task data
14	Total owned	1, 2 or more developers	✓	Task data
15	Estimated hours	No. of hours	✓	Task data
16	Daily meeting attended	Yes/no	✓	Progress data
17	Progress details	Yes/no	✓	Progress data

As described in Sect. 3.1, the case studies were used to demonstrate the ART Process Flow (Fig. 2) which contains essential stages in Input, Process and Output stage. The most vital part of the process is to determine its environment data and risk rules for the project. This is further elaborated on in the following sections.

4.2.1 The environment data

At the beginning of this work, two agile project management tools were studied; Extreme manager[1] and Rally software[2] in order to define possible environment data that may be available in a real-world scenario and could be used in this work. The environment data were categorized as follows:

- Project data—contains information about the project i.e. project name, start date and end date.
- Team data—contains information about the team members i.e. skills and experience.
- Task data—contains information on the user stories and breakdown of the tasks associated with the estimated hours.
- Progress data—contains information about the team reports on task progress.

After this information has been obtained from the studied tools, four categories of data described above were focused. Upon starting up the first case study, the ART template was set up for collecting data from these categories. The collected data from the student projects was as far as possible, screened and matched in order to meet the general cases from the studied tools. Even though the student projects did not use any of the studied tools the same environment data was available in their environment by other means. The summary of the possible environment data that can be collected is simplified in the following Table 2. Note that there are two available data items that were not used in this study; Total number of projects involved and also Team skills since the students in the case studies were only involved with one project at a time and all students fulfilled the required skill in programming, which in this case was a need for C# experience. It is envisaged that in an industrial project this data would also be used to identify risks.

4.2.2 The risk rules

In previous studies, the research problems and issues in agile projects were discussed and transformed into a set of problem scenarios which was then presented in Fig. 1. Each problem scenario represents a possible risk event that is associated with a Sprint goal for the project. The Sprint goal is important since it can be used to consider how environment data values could be used as indicators of threats to those goals i.e. triggers for the risks. Therefore, it is proposed that risk rules can be formulated using the risk indicators to identify events that cause loss (delay/extra cost/loss of value) i.e. risks, leading to a situation where risk identification can be automated.

Table 3 below show the sets of risk rules and risk indicators for the problem of task ownership. This problem is

[1] Hindsa, Extreme Manager, http://www.hindsa.com, accessed 14 March 2014.

[2] Rally, http://www.rallydev.com, accessed 14 March 2014.

Table 3 Rule template for task ownership

Goal	G1: In sprint X, task Y should be assigned to appropriate number of developers once Sprint X is started
Problem scenario	PB1: during the sprints, the developer does not have any pair or has too many programming partners for the selected task
Consequences	Avoiding 'single expert' or too many developers sharing code
Indicators	IN1.1: Project is started and when a task is selected in the sprint, 'task paired by' is empty—indicates high risk IN1.2: Project is started and the selected task owned by more than 2 developers—indicate low risk
Repository/data	Project data Task data
Rule(s)	RL1.1: If Project.Project_Status = [In Progress] AND If Task.Paired_By = [""] RL1.2: If Project.Project_Status = [In Progress] AND If Task.Total_Owned > 2
Risk ID and Name	RN1.1: R0001 pair programming RN1.2: R0002 task ownership

The remaining project goals; G2: Skills and Experience, G3: Resources and G4: Progress, are not discussed further in this paper. However, the generated risk rules associated with goals described earlier in Fig. 2 were mapped to goals as follows (Table 4). In each case the risk represents a threat to the goal success

Table 4 Mapping goals to associate Risk ID used in Case Study Alpha (CSA) and Case Study Beta (CSB)

Goals	Risk ID
G1: task ownership	R0001, R0002
G2: skills and experience	R0003, R0004, R0009, R0019
G3: resources	R0005, R0006
G4: progress	R0007, R0008

inferred from the risk issues related to people and using the Rule template presented in Table 1. Earlier, the ART GSRM model was developed that shows the relationship between goal, risk-obstacle and assessment. Using this model, the following rule template table (Table 3) show one of the sets of goals, problem scenarios, rules and risk indicators used in this work.

Table 3 indicates the set of rules and risk indicators for goal G1 where when a sprint is started, an appropriate number of developers should be assigned to the particular tasks. The usage of indicators generally depends on how the project manager wants to signify that the condition appears to be at risk. Two indicators were selected for the goal based on the risk issues highlighted earlier. Indicator IN1.1 states that once the Project Status was [In Progress] and the Task selected has no pair [" "], i.e. null or empty string, then the Risk 0001—"Pair programming" is triggered. IN1.2 states that once the Project Status is [In Progress] and the Task is owned by more than two developers [>2] then the Risk 0002, "Task ownership", is triggered. These indicators are then translated into rules RL1.1 and RL1.2 that contain object, attributes and value of the attributes that will be picked up by the agents. The repository section shows where the environment data is involved for the particular risks. As mentioned earlier, the set of indicators and rules can be updated from time to time depending how the project manager decides to identify risk. One such case is the modification of the rule for pair programming which took place between CSA and CSB.

4.3 The case studies

Given that the approach is novel and still at a research stage, the chosen study was rather exploratory in nature, where it relies on the collection of existing data used in the project as opposed to ongoing data collection in an ideal situation. Where possible, multiple sources of evidence (triangulation) were used, meaning that archival artefacts and informal interviews with the product owner were used to confirm findings. For example, after each collection of data an informal interview with the product owner was held in order to verify the interpretation based on the collected data. As mentioned earlier, students had no knowledge that their project data was being be assessed for risk, removing any possible bias in this respect. Rather they were motivated in demonstrating that they had followed agile project management practices e.g. pair programming as taught in classes and producing high quality working software. Based on the data collected in CSA, some issues were found in adopting the approach. These were noted along with conclusions of further discussions with the product owner. The outcome from this investigation was recorded in the investigation notes. Further, one modification was made to one rule, R0001, to be used in the next case study.

The following section discusses in detail the ART process flow (Fig. 2) as conducted in the two case studies. Both case studies consisted of two sprints so that the process flow was repeated iteratively in four instances.

Table 5 The environment data extracted from the student project artifacts

No.	File name	Data available/collected
1	Hartmann-Orona Spreadsheet	Sprint backlog information Major task area/user stories Task name Task owner Task status (completed/in progress/not started) Estimated hours Commits days and hours for each task Team member information Team member name and initials Working days for this sprint Working hours for this sprint Start date End date No. of calendar days Sprint team member daily activity Team burndown chart Team member burndown chart
2	Sprint backlog target	List of user stories Points for each user stories Dependencies
3	Scrum minutes of meeting (Daily)	Name of the scrum master for the day Work progress for each team member Work done since yesterday Work planned today Problems
4	TortoiseSVN repositories	Directory and files versioning Commit files/code Track changes
5	Source code (C#) integrates with resharper	Group project source code Code quality analysis

4.3.1 Case study alpha (CSA)

As described earlier, the ART Process Flow consists of three main stages; input, process and output. In this section this process is described in detail.

The Input stage starts by gathering all necessary data from the student project artefacts and translating the data to match the ART Template. During the Input stage, there were two inputs needed—definition of the list of environment data and definition of the risk rules used in this study. In order to define the list of environment data, two steps were performed.

1. Gathering data

For the purpose of defining the list of environment data, archived artefacts derived from the student projects were used. There were five main artefacts used in this study, summarised as below.

- Hartmann-Orona Spreadsheet—a well known spreadsheet mostly containing sprint backlog data, including the breakdown of each user story into tasks, estimated hours and owner of each task.
- Sprint Backlog Target—contains a list of user stories and associated points and dependencies.
- Scrum Minutes of Meeting—contains information on team attendance and updates on tasks.
- Code Repositories—a subversion (http://subversion.tigris.org) source control system that was used by students to manage their project. Each group was required to log their activities and check in all documents and source code.
- Source Code—the students were required to use the C# and VB.Net programming languages

Table 5 below shows the list of extracted data from the archived artefacts of the environment data.

The archived artefacts available however, did not provide as much data as the studied commercial tools. Nevertheless, the archived artefacts contained enough useful information, particularly related to sprint backlog and the user stories, breakdown of the tasks, details of the developer responsible for a task and so on. In addition, the goal of the study was to demonstrate the approach and tool support, not applicability to every data item collected in mainstream tools.

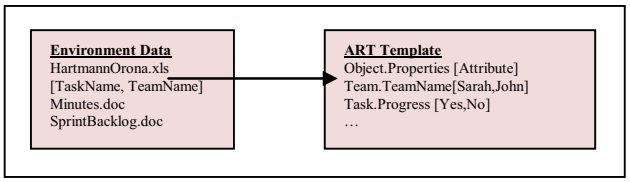


Fig. 7 Translating environment data to ART template

One issue that had to be overcome was that of missing data in the SVN repository. Although all student groups were required to log their work into the repository, some groups had not done this. For example, all groups were required to record daily minutes of meetings yet some of the minutes were missing from the repositories. Consequently, the product owner had to trace this record through other methods such as email archives to obtain the missing data. Similarly, there were some inconsistencies in the format of the minutes of meetings. For example each group should have specified the name of the Scrum Master for each meeting. However, this information was missing in some of the groups. Again, the product owner had to retrieve through email archives and provide this information. All issues found were recorded and written in the investigation notes so that the process could be improved in the future.

2. ART data translation

Once all data had been defined and organized into its categories, the next step was to translate this data manually to fit the ART template. Figure 7 shows an example of the translation of the data from the archived artefacts to the ART template. This includes transforming the raw data obtained from the artefacts in the form of object oriented concepts.

This is essential so that the ART agents will be able to pick up the data and match this with the rules embedded in the tool. Since this study was done after the project had ended, the data obtained was comprehensive starting from day one in sprint SP1 until day 15 in sprint SP2 and ready to be translated into the template. In the event where the project is new or 'fresh', data can be keyed or added directly through the user interface. Similarly, there were also issues highlighted while doing this step. The main issue identified was associated with the process of translating the raw data into the template. Since this was done manually, it involved tedious and highly effort intensive tasks. In this case study, there were six student groups with six different sets of archived data. Even though they might have the same format or almost the same format of the document, the interpretation and explanation of the data was different. To overcome this it was often necessary to carefully go through all the documents in the repositories in order to understand how they implemented their project. Additionally, the problem is compounded in the case of tracking a task assigned to a team member. For example, in the Hartmann-Orona spreadsheet, the team defined the user stories and the breakdown of the tasks. The spreadsheet itself did not provide a unique id for each of the tasks although it did include the name of the person responsible for the task. Whilst in the team minutes for each meeting each team member provided updates on the task they were assigned, this was only briefly described. In the event where which team member committed to a specific task had to be identified and were tallied with the spreadsheet,

Table 6 List of risk name along with its associated rules and probability and impact score

Risk ID	Risk name	Rules [Object.Attribute] = [Value]	Prob score	Imp score
R0001	Pair programming	PROJECT.PROJECT_STATUS = Completed TASK.PAIRED_BY = ""	3	5
R0002	Task ownership	PROJECT.PROJECT_ STATUS = Completed TASK.TOTAL_OWNED > 2	1	1
R0003	High priority task assigned to inappropriate team member cannot be completed on time	TASK.PRIORITY = High TEAM.SKILLLEVEL = 1	5	4
R0005	Overload tasks can cause difficulty in time management	PROJECT.PROJECT_STATUS = Completed TEAM.TOTAL_NO_ROLE > 1	5	1
R0007	Developer absent in meeting possible of employee turnover	PROJECT.PROJECT_STATUS = Completed PROGRESS.DAILY_MEETING = N	1	3
R0008	No progress report	PROJECT.PROJECT_STATUS = Completed PROGRESS.PROGRESS_DETAILS = N	1	3
R0009	Unable to understand agile process and meet the target	PROJECT.PROJECT_STATUS = Completed TEAM.AGILE = false	3	3
R0019	Unable to comply with the agile process	TEAM.AGILE = true TEAM.EXPERIENCE = Very Poor	1	1

there is a need to go through the repositories and look at the code commits by the team member. The time consumed was 2–3 h for translation of one group's data, excluding time spent retrieving missing data.

Next, the risk rules for this case study were defined. Table 6 below shows the list of risk name and rules as discussed at the beginning of this chapter as well as its probability and impact score as defined for each risk. Note that the rules were embedded in the Rule engine during the development of the ART prototype tool and at this stage existing rules from the Rule engine database had only to be selected. However, when needed, new rules were added into the Rule engine or existing rules edited using the provided user interface. Similarly, when entering the probability and impact the parameters could be adjusted later on. For this case, the value of the probability and impact score for each risk was cross checked with the Product Owner. Since this was a student project, there was no actual impact on cost involved for this project. Therefore, the impact factor was based on the consequences of the student not completing the project and not producing a quality end product as required by the Product Owner. In the real world project, the risk identified will be more project-specific, in other words risks are assessed individually for a specific project situation or environment. A significant project risk can be the result of certain characteristics in the project environment. For example, a developer who is considered to have very low skills but is assigned to a high priority task could lead to a higher risk exposure compared to where that developer is assigned to a lower priority task. In brief, the project manager determines what risk is significant and how severe the risk is.

The Process stage is the stage where the risk assessment automatically took place. Based on the defined inputs described previously, the ART agents communicate between the ART template and the Rule engine. At this stage, once the project is loaded into the ART prototype, changes can be made using the provided user interface. Once the tool is 'run', the ART agents will react if any of the rules are triggered and then notify an identified risk. Any changes in the inputs will result in changes in the outcome of the triggered risk as well. This is because the identified risk can be observed in the Output stage and the problem of ignoring a risk is avoided.

In the Output stage, once a risk is identified, the risk result is displayed in the Risk Register. This should also show an overview of all risks triggered. This includes the risk name, location of the risk associated with the affected task and the owner of the risk, defaulted to the owner of the task. The risk register also displayed the risk result according to priority, starting from the risk with high to low severity. After one sprint is completed, the risk result is stored into the Risk data repository.

After each sprint in CSA was completed, reports were created. The presented reports provide useful information on the total of risk score each day and in one sprint. This includes information on the breakdown of risk identified each day.

4.3.2 Case study beta (CSB)

Based on the experience from CSA, investigation notes revealed the following issues:

1. Design of the project: Since the project was designed for students as part of a university course, the real goal in practice was for students to apply what they had learnt during the course. Hence changing the structure of the project was not possible. Further, due to the limitation of time in completing their project it is not possible to add more management tasks. Nonetheless, data collection needed to be easier.
2. Format of the document: In order to avoid missing data, standard formats were established for documents used for data collection in the project; i.e. meeting minutes.
3. Naming conventions/traceability: It was decided that in order to easily track the data between a task and its owner a unique id for each task and each team member was required. For example, all tasks should start with a unique id beginning with "TS" e.g. TS001 and all team members could have a unique id starting with "TM" e.g. TM001.
4. Task allocation and estimation: Based on the summary of data collected on each group project background, it is found that some groups failed to allocate tasks evenly to each team member. Some team members carried out too many tasks which resulted in some of the tasks not being completed. Further, it was noticeable that some of the estimated task sizes were too big and should have been split into smaller tasks of almost the same size. It was also emphasized that team members should practice pair programming whenever a bigger task is assigned.
5. Specifying student skill level and agile experience: During sprint SP1 it was proposed that student skill and agile experience should be taken into account. At this stage, identifying student skill was relatively easy. Specifying agile experience is more problematic. In SP1 it is assumed all students did not possess any agile experience but their agile experience was then measured in SP2 based on the assessment by the educator from the first sprint.
6. Risk rule for pair programming: Risk results were presented to the product owner and one of the most common risks found was related to absence of pair pro-

Table 7 Risk name along with its associated rules and probability and impact score (modified)

Risk ID	Risk name	Rules [Object.Attribute] = [Value]	Prob score	Imp score
R0001	Pair programming	PROJECT.PROJECT_STATUS = Completed TASK.PAIRED_BY = "" TASK.ESTIMATE > 5	**3**	**5**

gramming. It was obvious, from the findings that most of the students did not adhere to this practice. However, the argument was that some of the tasks were too small and were not suitable for work in pairs. As such, it is essential to propose to modify the rule, where the modification being described further in the following case study—CSB.

Based on the lessons learnt, the improvements to data collection were made and to performance of the students in applying the agile practices learnt in their course. Students were informed of these. At this stage, it is assumed that the product owner's awareness of the students' performance has increased based on the lessons learnt from CSA. In addition, it is expected to change the performance of the student group in this case study.

During the Input stage, the same steps included in the previous case study were adhered. As described previously, due to the nature of the project we were not able to change the structure of the project and therefore the same artefacts were used and data capture methods from these artefacts.

At the data gathering step the process was found to be much easier than in the previous case study. For example, some of the student groups used the new naming convention in naming their reports and one group used the format of the proposed minutes of meeting. As reported earlier regarding some missing data from the previous student projects and for this case study there was one group who had not logged their main document which was the Hartmann-Orona spreadsheet in the repositories. This document was untraceable therefore there is a need to drop this group from this study. Although the acceptance in changing the working method is rather poor in this instance this can be improved gradually. Similarly, in a real world project it is normal that some organizations might refuse or feel challenged when asked to change their methods. However, due to the limitation discussed earlier it is not possible to compel the students to adhere to standards due to the constraints of this also being an assessment exercise.

As discussed earlier the difficulty in carrying out the process of transforming the raw data into the template during the ART data translation step. Although the effort of performing this step is still quite intensive, the time to translate one group's data was reduced to less than an hour. This was especially the case where the standard naming convention was used in reporting for some groups. In addition, it is agreed that to implement this approach the first time was rather challenging. This was greatly improved and the process would be more effortlessly managed if adopted repeatedly.

Next, the risk rules for this case study were applied. In the previous case study, the list of the risks and their associated rules (Table 6) were summarized. Based on the outcomes and lessons learnt from the previous case study, the pair programming rule was modified in order to provide a more realistic risk rule. The ability to modify the rules as needed demonstrate that the solution approach and tool support is dynamically responsive to changes thus, as is required in agile projects. This is described in Table 7 which shows the highlighted risk rule as modified. The remaining risk rules were unchanged.

Once the Process and Output stage have been completed for the two sprints, again, the reports on the information gathered on the Risk data were created and presented in the form of diagrams. The outcome of both case studies has yielded results of Total Risk Score (TRS).

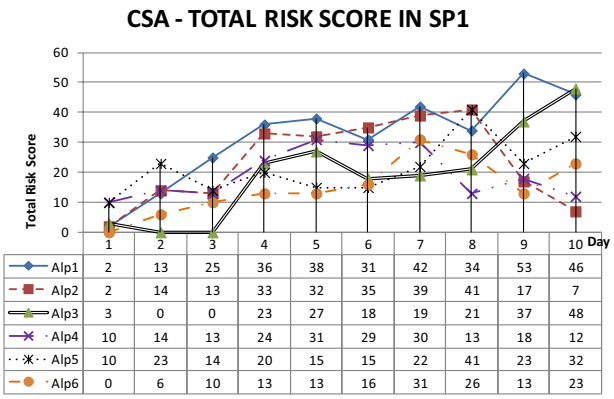
4.4 Case study results

Risk data derived using the tool and displayed in a Risk Register was recorded and saved in the Risk Data Repository. This risk results later were assessed and presented using graphs.

4.4.1 Summary of CSA project data

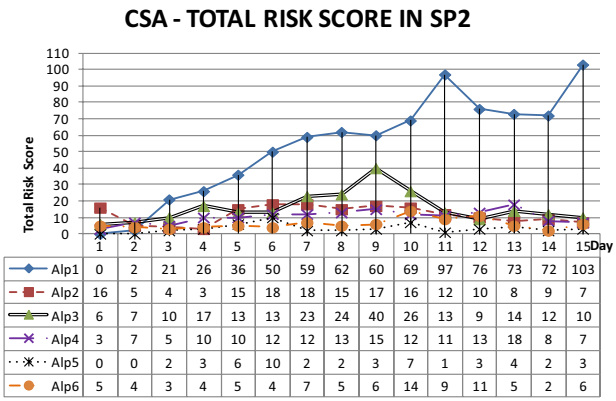
During the planning phase, each team was given Product backlog items that consisted of a list of user stories. They were asked to estimate the work effort needed for each user story in hours and break this down into a set of tasks. Hence, each team had a varying range of total number of tasks in each sprint. At this stage, the students were challenged, based on lectures taught in class, in their ability to plan and estimate the work effort required for each sprint.

The output from the tool is used as means of assessing the total risk in the project at any point or in a post sprint review as shown in Figs. 4 and 5. This graph includes information on the breakdown of risk identified each day using total risk score (TRS). TRS is based on the generic severity score of a risk item for the task it is related to. The metric provides results on counting of risk daily and cumulative



	1	2	3	4	5	6	7	8	9	10
Alp1	2	13	25	36	38	31	42	34	53	46
Alp2	2	14	13	33	32	35	39	41	17	7
Alp3	3	0	0	23	27	18	19	21	37	48
Alp4	10	14	13	24	31	29	30	13	18	12
Alp5	10	23	14	20	15	15	22	41	23	32
Alp6	0	6	10	13	13	16	31	26	13	23

Fig. 8 Total risk score graph of case study alpha (Sprint 1)



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alp1	0	2	21	26	36	50	59	62	60	69	97	76	73	72	103
Alp2	16	5	4	3	15	18	18	15	17	16	12	10	8	9	7
Alp3	6	7	10	17	13	13	23	24	40	26	13	9	14	12	10
Alp4	3	7	5	10	10	12	12	13	15	12	11	13	18	8	7
Alp5	0	0	2	3	6	10	2	2	3	7	1	3	4	2	3
Alp6	5	4	3	4	5	4	7	5	6	14	9	11	5	2	6

Fig. 9 Total risk score graph of case study alpha (Sprint 2)

risk counting in a sprint for continuous risk management purposes. The TRS is calculated as below.

Consider that there is a set of tasks T in the project on a given day d:

$T_d = \{t_1 \ldots t_n\}$ where n = total number of tasks on day d.

There is also a set of predefined generic risks R that can potentially be identified in the project:

$R = \{r_1 \ldots r_m\}$ where m = total number predefined risks.

Thus, risks associated with each task, t on a given day d is $R_{td} \subset R$.

The TRS for a task t on day d is therefore.

TRS_{td} = card (R_{td}), where card is the cardinality of the set.

These risks can be associated with any of the tasks t in T. The risks are present while the task is being carried out in the sprint.

Therefore, for a given day d the TRS is

$$TRS_d = \sum_{t=1}^{t=n} TRS_{td},$$

where the total of risks triggered will be calculated in all tasks carried out in that particular day.

The application of TRS can be used in a current or ongoing project or a past project. In a current project, TRS is calculated daily. This is when the project manager can see the risk triggers from day one and if it is resolved, the risk is no longer appears. On the other hand, TRS can be applied to a past project as a means for review. The risk data obtained can be used as an input using the proposed approach or to predict risk for a future project.

Figures 8 and 9 below show a plot of the TRS for both sprints. In SP1, the number of risks score ranged from as low as zero (either no risk being present or no ongoing task) up to the highest of 53 risks found in team Alp1, Day 9 of the sprint. In SP2, risk score ranged from zero to the highest of 103, also found in team Alp1, Day 15. Both

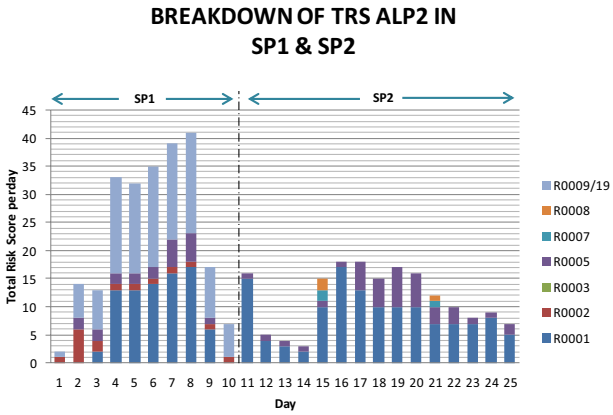
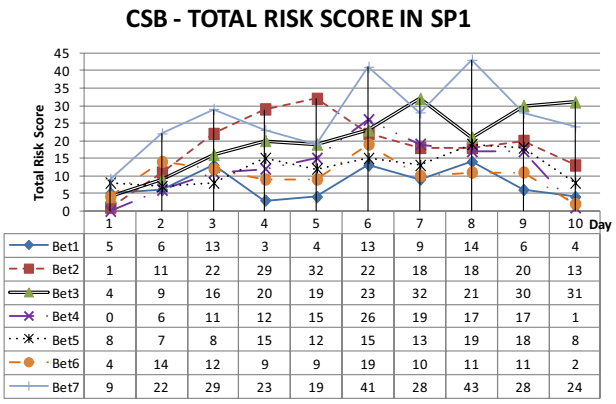


Fig. 10 Graph showing the breakdown of Total Risk Score for team Alp2 in SP1 and SP2

graphs show the increasing and decreasing pattern of risk detected for each team with some peaks at certain days which give a better visualization of risk.

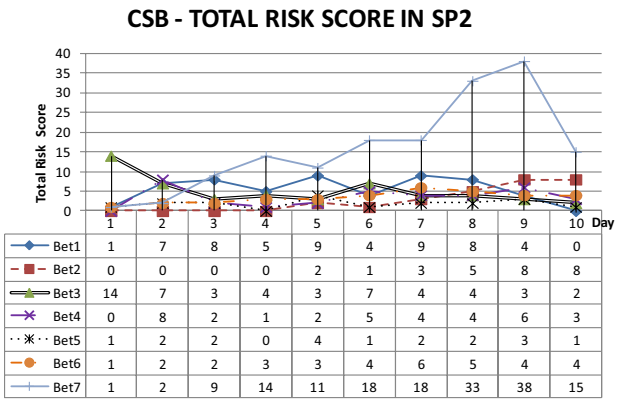
The already-established risk burndown chart (Cohn 2010) generally results in a downward risk exposure graph, computed from the changing probability and size of potential losses in every sprint. However, that technique does not show or visualize specifically which risk is being identified, assessed and monitored throughout the project. On the other hand, the graph below may be used to develop a risk trend for a type or a category for agile team. For example, a team who are new to agile projects, one might see risks might increase gradually to a peak and start decreasing towards the end of the sprint as risks are resolved or tasks completed. On the other hand, a more established team might not have any risk occurring daily in their sprint; instead showing a peak at certain times with regards to the type of risk that they want to monitor.

The data shown in Fig. 10 provides a better visualization of the type of risk triggered each day in team Alp2, for both



	1	2	3	4	5	6	7	8	9	10
Bet1	5	6	13	3	4	13	9	14	6	4
Bet2	1	11	22	29	32	22	18	18	20	13
Bet3	4	9	16	20	19	23	32	21	30	31
Bet4	0	6	11	12	15	26	19	17	17	1
Bet5	8	7	8	15	12	15	13	19	18	8
Bet6	4	14	12	9	9	19	10	11	11	2
Bet7	9	22	29	23	19	41	28	43	28	24

Fig. 11 Total risk score graph of case study beta (Sprint 1)



	1	2	3	4	5	6	7	8	9	10
Bet1	1	7	8	5	9	4	9	8	4	0
Bet2	0	0	0	0	2	1	3	5	8	8
Bet3	14	7	3	4	3	7	4	4	3	2
Bet4	0	8	2	1	2	5	4	4	6	3
Bet5	1	2	2	0	4	1	2	2	3	1
Bet6	1	2	2	3	3	4	6	5	4	4
Bet7	1	2	9	14	11	18	18	33	38	15

Fig. 12 Total risk score graph of case study beta (Sprint 2)

sprints SP1 and SP2. The advantage of this method was twofold; (1) one would be able to view the category of risk triggered for the particular day as well as (2) one would be able to view which risk being triggered the most and needs attention from the project manager in order to help in making decisions e.g. whether prompt action should be taken towards the risk that occurred consecutively. For example, in SP1, team Alp2 had generated risk mainly through not performing pair programming and from having team members without agile experience. However, in SP2, since the students had already developed using agile in SP1, the risk of no agile experience no longer existed, but the risk from not performing pair programming could still occur. In the student projects these risks if realized might have very little impact towards project success but in the real-world project and these risks are triggered without being addressed a threat to the project could arise. On the other hand, patterns of risk occurrence for a particular team member can be inferred and used for estimating future project risks in similar projects with the same team members.

4.4.2 Summary of CSB project data

Comparing with the TRS presented in the CSA earlier (Figs. 8, 9) for both sprints, the TRS for CSB was much lower than the TRS in CSA. This has supported the assumption stated earlier that the indication of possible inspection with regard to risk may affect the developer or student behaviour during the project. Despite the modification of the pair programming rule for CSB, the TRS produced in this case study was more realistic so that pair programming was critical only in bigger tasks. This may be due to the fact student projects were used in the case study and a less motivation to do pair programming than if told to in a workplace. Thus, a high number of risks have arisen due to the violation of this rule. In practice, the project manager could modify this rule from time to time.

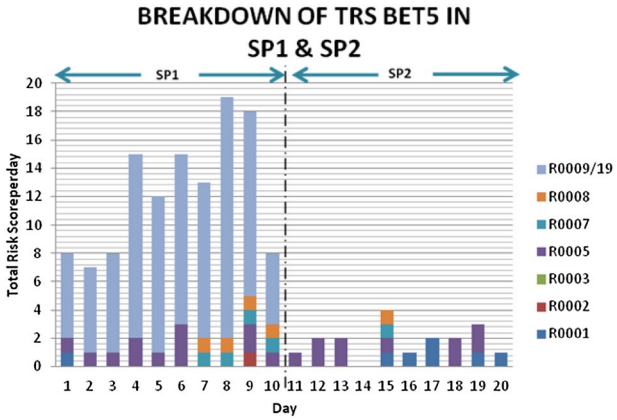


Fig. 13 Graph showing the breakdown of total risk score for team Bet5 in SP1 and SP2

Figures 11 and 12 show the calculated TRS for both sprints. In SP1, the risk score ranged from as low as zero (either no risk being present or no ongoing tasks) up to the highest risk score of 43 in team Bet7, Day 8 of the sprint. Whereas in SP2, the risk score ranged from zero to the highest risk score of 38, which was also found in team Bet7, Day 9 of the sprint. Both graphs show the increasing and decreasing pattern of risk detected for each team with some peaks on certain days thus providing us a better visualization of the risks arising. Normally one would expect risk should decrease (burn down) over a sprint. However, referring to both figures of a sprint review, for Team Bet1 on both sprints would demonstrate clearly a useful outcome of using the tool, that the team performance here was problematic.

Similarly to CSA, in the light of viewing specific risks that triggered on a particular day, the data shown in Fig. 13 provides better visualization of the type of risk triggered each day for team Bet5, for both sprints. The advantage of having this method is that we would be able to view the

type of risk triggered for the particular day as well as to view which risk is being triggered the most and therefore which should be countered as a priority.

5 Discussion

The results obtained from the case study have offered evidence that apart from the novelty of the proposed approach and tool support, the approach is usable and provides useful data. The TRS graphs produced in both case studies provide a useful visualization method which supports identification and monitoring of risks in a dynamic agile project. They show the number of risks picked up by the ART agents and provide a realistic and interactive way of monitoring risks. The study revealed that it is possible to use existing SE environment data to support risk management. However, there are many environment data items that can be used to detect risk thus, acquiring project manager to define which ones that are related to their project. Further to this, our empirical evidence also revealed that data can be collected with minimal intrusion and effort.

Three Research Questions (RQs) were established earlier as an expression of the aims of this research work. The following paragraphs summarize how these have been responded to in the case studies.

RQ1: The conclusion provides evidence that it is possible to use the project environment data to identify risk and so overcome the main barriers in the application of risk management. This supports the notion that, in agile processes, a lightweight risk management approach is required that automates some of the risk processes. As a result of this, a solution was developed using software agents to react to the project environment, based on designated rules in order to manage risk.

RQ2: The method used for both case studies evidencing that data collection conducted from both case studies involved minimal intrusion and effort, and with no cost involved. In the cases studies employed, the environment data used includes the student project data, in this instance from archived data. The data was stored in SVN repositories and was retrieved by the educator for use in the case study. Both case studies did not include any interaction with the participants in the study setting in order to meet the purpose of the study. This includes to identify risk in their project and to investigate compliance of the team member with agile practices. The collected data was validated by the educator/product owner of the student's project based on discussion sessions prior to and after the implementation of the project.

RQ3: A prototype tool was developed in order to validate the interaction between agents, agents' compliance with the designated rules and how agents react to changes in project environment data. This is discussed throughout Sect. 3.1.1 starting from defining the input, processing the input and producing the output. Later, a walkthrough of the ART process is adopted in both case studies supported by the prototype tool. This demonstrates that software agents coupled with a rule engine can automate risk management using data from the project environment.

5.1 Study validity

Since the study presented introduces a new approach in managing risk in agile project, the main issues are focused on the internal threats. The first internal threat is in terms of the accuracy of the measured data, especially because the data used was based on historical artefacts. Further, confirmation of this data was not possible as the project had already been completed at the time of analysis. Secondly, the approach used entailed manual collection and translation of data from archived artefacts into the ART tool. This human effort was required before the ART agents could begin reacting towards environment data as they were designed to work in. This effort could be minimized by selecting a proper individual in the team to conduct this process, for example the Scrum Master in a Scrum project.

One step taken to ensure the quality of the study was that cross checking was done from time to time with the Product owner to confirm perception based on his observation. Considering external validity threats, the risk management approach and tool supports were designed to be as general as possible so that this is applicable in general to agile project environments. This includes taking into account two popular agile project management tools studied for this work so that the approach is as applicable as possible to other contexts but also lightweight and unobtrusive to the team daily activities. Nonetheless, no claim can be made of good fit with tools not studied. Additionally, the study used student project data along with the case study execution guideline (Runeson and Höst 2009) rather than industrial data. Hence, there will be arguments whether this is applicable to a real world environment.

6 Conclusion

In this paper we presented a novel approach to manage risk in agile projects. The work offers contributions in two areas (1) on the use of case studies for assessing new methods and tools and provides an example of how student teams can be used to gather information not feasible in industrial settings. (2) on the use of agents to semi-autonomic ally manage software risk.

This work provides several significant investigations on the problems and issues in risk management specifically in

agile projects. The development of the ART model and tool support has been demonstrated to help by at least reducing the problems previously identified with risk management. The approach is necessarily supported by a prototype tool which has been shown to manage risks in example agile projects. The role of risk management in iterative and agile processes has to date been neglected but this model integrates risk management model with agile methods in a way that does not bloat the agile process.

This approach however, to the authors' knowledge and understanding has never been applied in risk management, especially with the specific aim of reduction of human effort. In addition, the resulting risk management process is naturally lightweight since each software agent is design to achieve a designated goal i.e. to identify, assess, prioritize or monitor risk. This paper has led to use designated software agents to facilitate the risk management process. Therefore, this work demonstrates the potential of autonomous computing being applied to risk management where software agents have been used to assist the human oriented and complex risk management process. In future, this work aimed to comprehend the physical implementation of the ART model and tool support, where there is a need to integrate this with existing Agile Project Management tools, perhaps as a plug-in, so that automated risk management can be fully realised. This would allow more practical risk management while a project runs in the foreground, software agents are in the background ready to manage emerging risks.

Acknowledgements This research has been funded partly by Universiti Teknologi MARA Malaysia and has been part of the Ph.D thesis work submitted to Queen's University Belfast, United Kingdom. This paper has been selected among best papers presented at the First EAI International Conference on Computer Science and Engineering (COMPSE) 2016.

References

Ahmed A, Kayis B, Amornsawadwatana S (2007) A review of techniques for risk management in projects. Benchmarking: Int J 14(1):22–36

Bandyopadhyay K, Mykytyn PP, Mykytyn K (1999) A framework for integrated risk management in information technology. Management Decision 37(5):437–445

Bannerman PL (2008) Risk and risk management in software projects: a reassessment. J Syst Softw 81(12):2118–2133

Bellifemine FL, Caire G, Greenwood D (2007) Developing multi-agent systems with JADE, Wiley.com

Boehm BW (1989) Tutorial: software risk management. IEEE Computer Society Press, Washington

Boehm B, Turner R (2003) Using risk to balance agile and plan-driven methods. Computer 36(6):57–66

Boehm B, Turner R (2005) Management challenges to implementing agile processes in traditional development organizations. Softw IEEE 22(5):30–39

Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J (2002) Modeling early requirements in Tropos: a transformation based approach. In: Agent-Oriented Software Engineering II. Springer, Berlin, pp 151–168

Chakradhar K (2009) Risk management in agile development. White paper edn. Polaris Software Lab Limited

Cho J (2008) Issues and Challenges of agile software development with SCRUM. Issues Inf Syst 9(2):188–195

Cockburn A, Highsmith J (2001) Agile software development, the people factor. Computer 34(11):131–133

Cohn M (2005) Agile estimating and planning. Pearson Education, London

Cohn M (2010) Managing Risk on Agile Projects with the Risk Burndown Chart

Conboy K, Coyle S, Wang X, Pikkarainen M (2010) People over process: key people challenges in agile development. IEEE Software

Dardenne A, Van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Programm (20)1:3–50

Deemer P, Benefield G, Larman C, Vodde B (2010) The scrum primer. Scrum primer is an in-depth introduction to the theory and practice of Scrum, albeit primarily from a software development perspective, vol 1285931497. http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf

Easterbrook S, Singer J, Storey M, Damian D (2008) Selecting empirical methods for software engineering research. In: Guide to advanced empirical software engineering, Springer, Berlin, pp 285–311

Fowler M, Highsmith J (2001) The agile manifesto. Softw Dev 9(8):28–35

Hindsa, Extreme Manager, http://www.hindsa.com. Accessed 14 March 2014

Hossain E, Babar MA, Paik H, Verner J (2009) Risk identification and mitigation processes for using Scrum in global software development: A conceptual framework. Software Engineering Conference, 2009. APSEC'09, Asia-Pacific, IEEE, p 457

Ibbs CW, Kwak YH (2000) Assessing project management maturity. Project Management Journal 31(1):32–43

Islam S (2009) Software development risk management model: a goal driven approach. In: Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium, ACM, p 5

Kitchenham B, Pickard L, Pfleeger SL (1995) Case studies for method and tool evaluation. Softw IEEE 12(4):52–62

Kontio J (1997) The RISKIT method for software risk management, version 1.00. Computer Science Technical Reports, University of Maryland, College Park, MD, USA

Layman L, Williams L, Cunningham L (2006) Motivations and measurements in an agile case study. J Syst Archit 52(11):654–667

Lindvall M, Basili V, Boehm B, Costa P, Dangle K, Shull F, Tesoriero R, Williams L, Zelkowitz M (2002) Empirical findings in agile methods. In: Extreme Programming and Agile Methods—XP/Agile Springer, pp 197–207

Melnik G, Maurer F (2006) Comparative analysis of job satisfaction in agile and non-agile software development teams. In: Extreme Programming and Agile Processes in Software Engineering Springer, pp 32–42

Melo C, Cruzes DS, Kon F, Conradi R (2011) Agile team perceptions of productivity factors. Agile Conference (AGILE), 2011 IEEE, pp 57

Menzies T, Williams S, Elrawas O, Baker D, Boehm B, Hihn J, LumK, Madachy R (2009) Accurate estimates without local data? Softw Process Improv Pract 14(4):213–225

Nelson CR, Taran G, de Lascurain Hinojosa L (2008) Explicit risk management In: International Conference on Agile Processes and Extreme Programming in Software Engineering. Springer, pp 190–201

Nerur S, Mahapatra R, Mangalaraj G (2005) Challenges of migrating to agile methodologies. Commun ACM 48(5):72–78

Nyfjord J, Kajko-Mattsson M (2008) Integrating risk management with software development: State of practice. In: Proceedings, IAENG International Conference on Software Engineering. BrownWalker Press, Boca Raton Citeseer

Odzaly EE, Greer D, Sage P (2009) Software risk management barriers: An empirical study. Empirical Software Engineering and Measurement. ESEM 2009 pp 418–421

Odzaly EE, Greer D, Stewart D (2014) Lightweight Risk Management in Agile Projects. In: SEKE (pp 576–581)

Patterson FD, Neailey K (2002) A risk register database system to aid the management of project risk. Int J Project Manage 20(5):365–374

Pfleeger SL (2000) Risky business: what we have yet to learn about risk management. J Syst Softw 53(3):265–273

Rally, http://www.rallydev.com. Accessed 14 Mar 2014

Ropponen J, Lyytinen K (1997) Can software risk management improve system development: an exploratory study. Eur J Inf Syst 6(1):41–50

Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131–164

Willams TM (1994) Using a risk register to integrate risk management in project definition. Int J Project Manage12(1):17–22

Williams RC, Walker JA, Dorofee AJ (1997) Putting risk management into practice. IEEE Soft 14(3):75–82